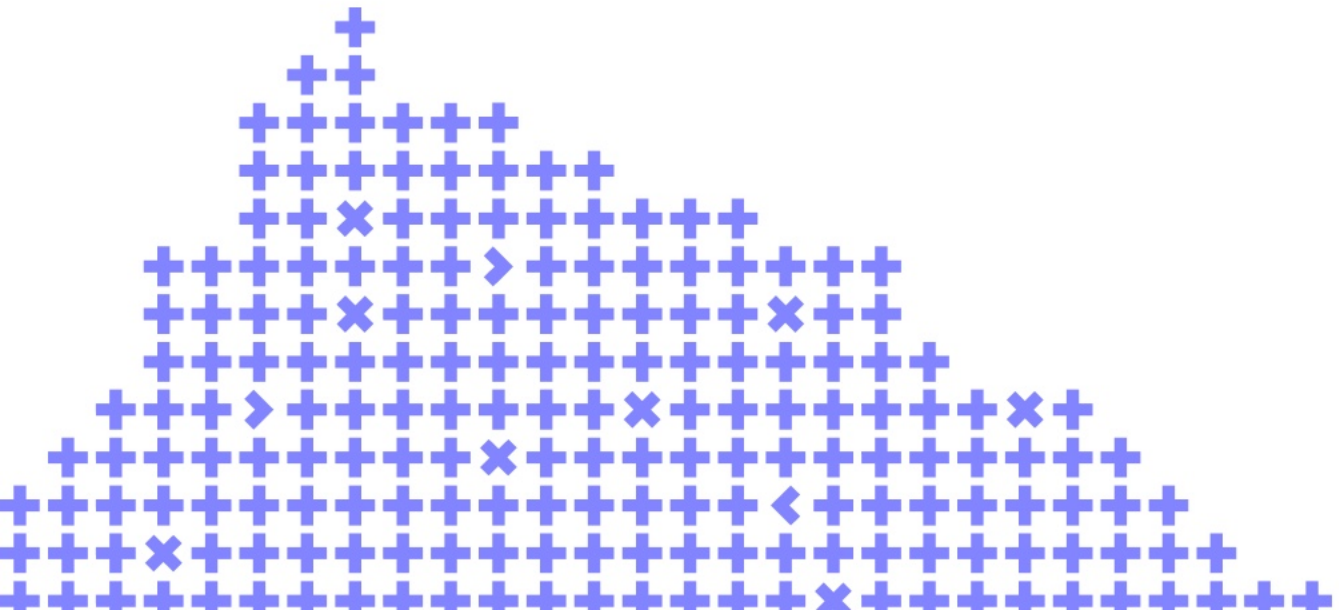
Kafka for Golang developers: tips and tricks

Denis Filippov



High Load Armenia

Co-organizer

Yandex

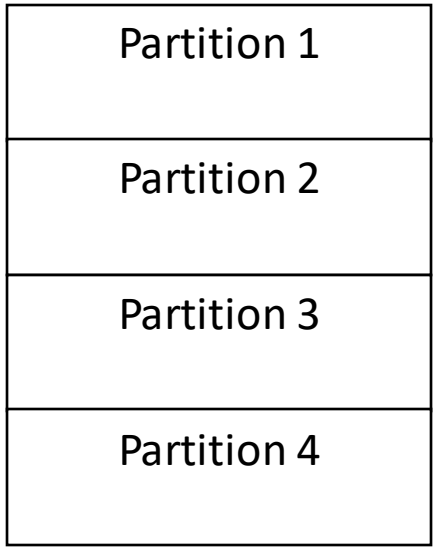
Your background

- It is not Kafka 101, so you have some basic Kafka experience
- Golang experience is not necessary

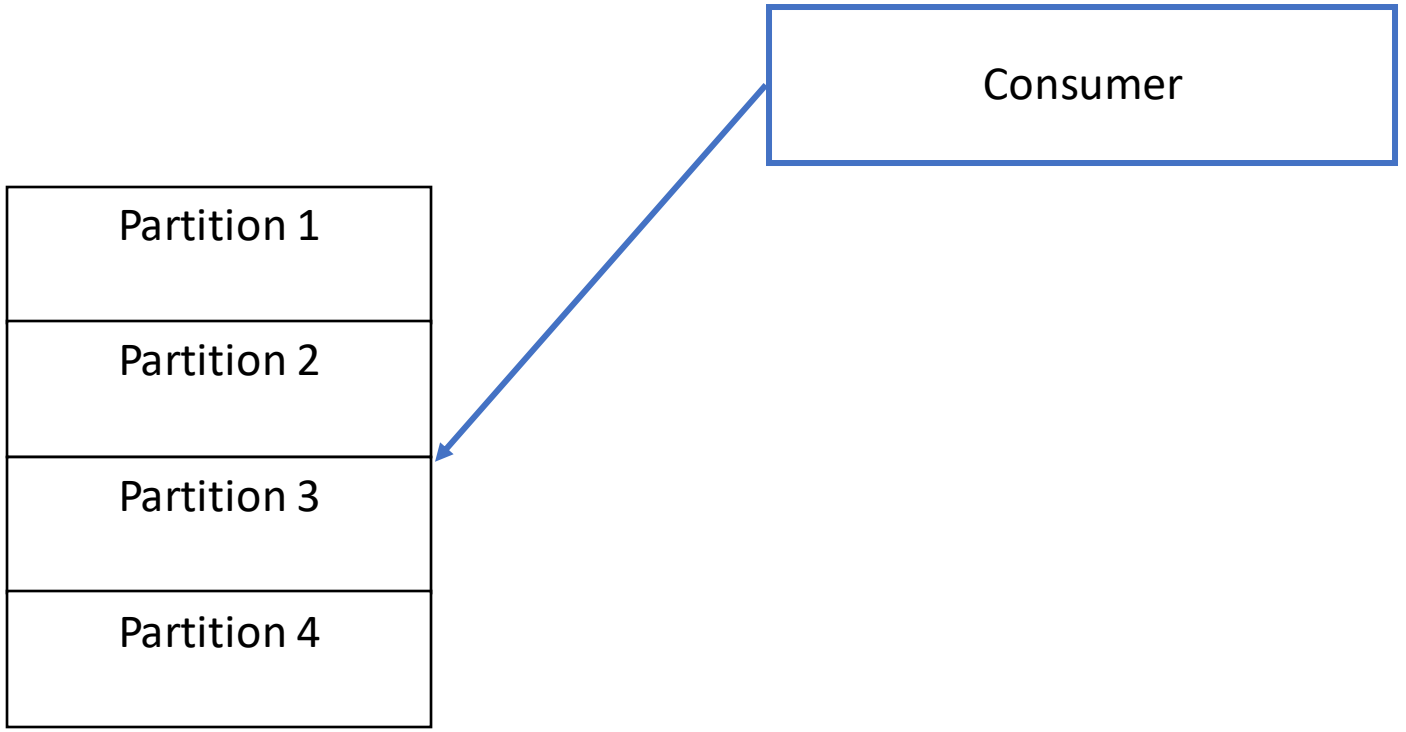
AGENDA

- Basics: partitions, consumers, consumer groups
- Rebalancing
- Manual commit
- Producing in batches

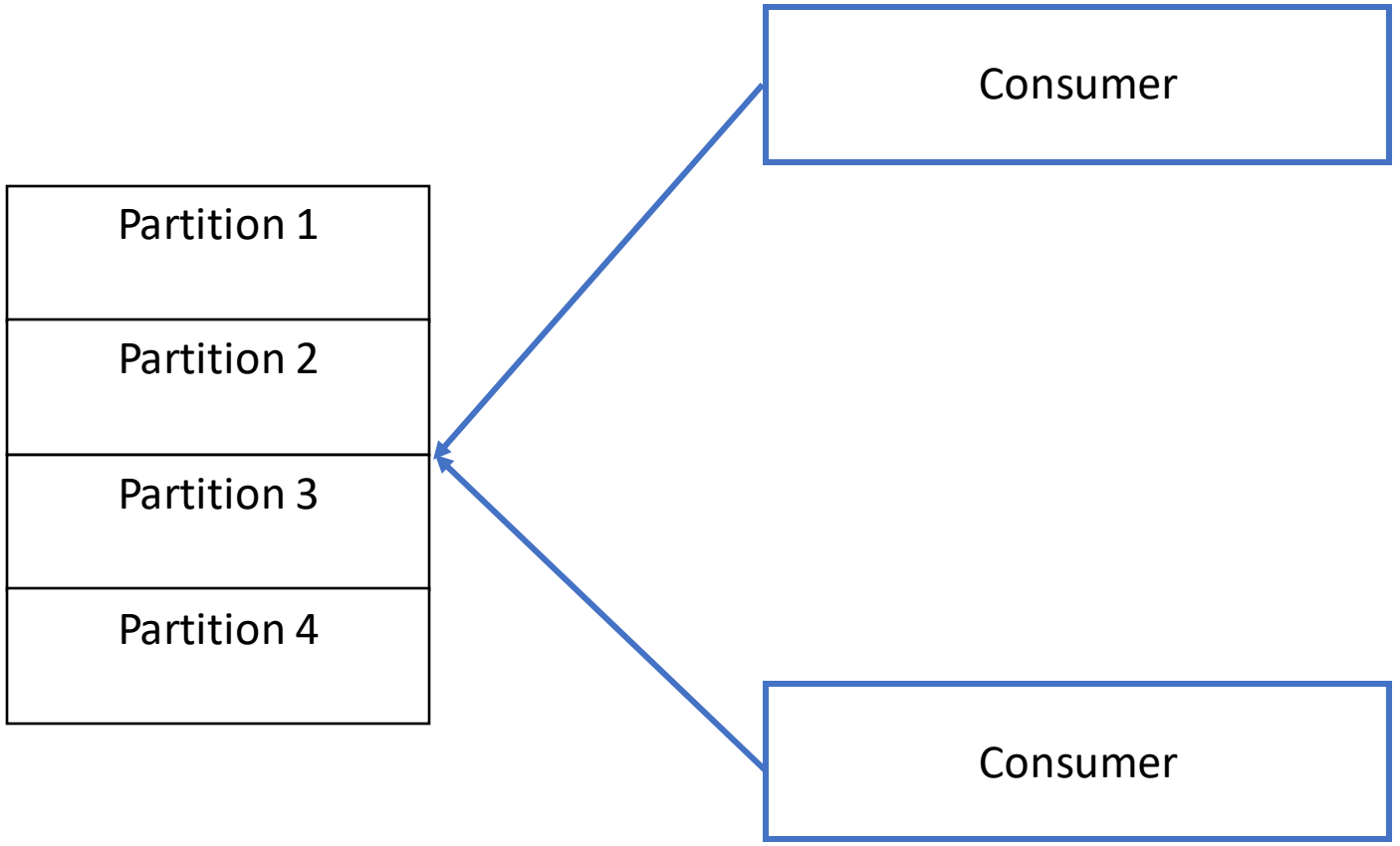
Consumers



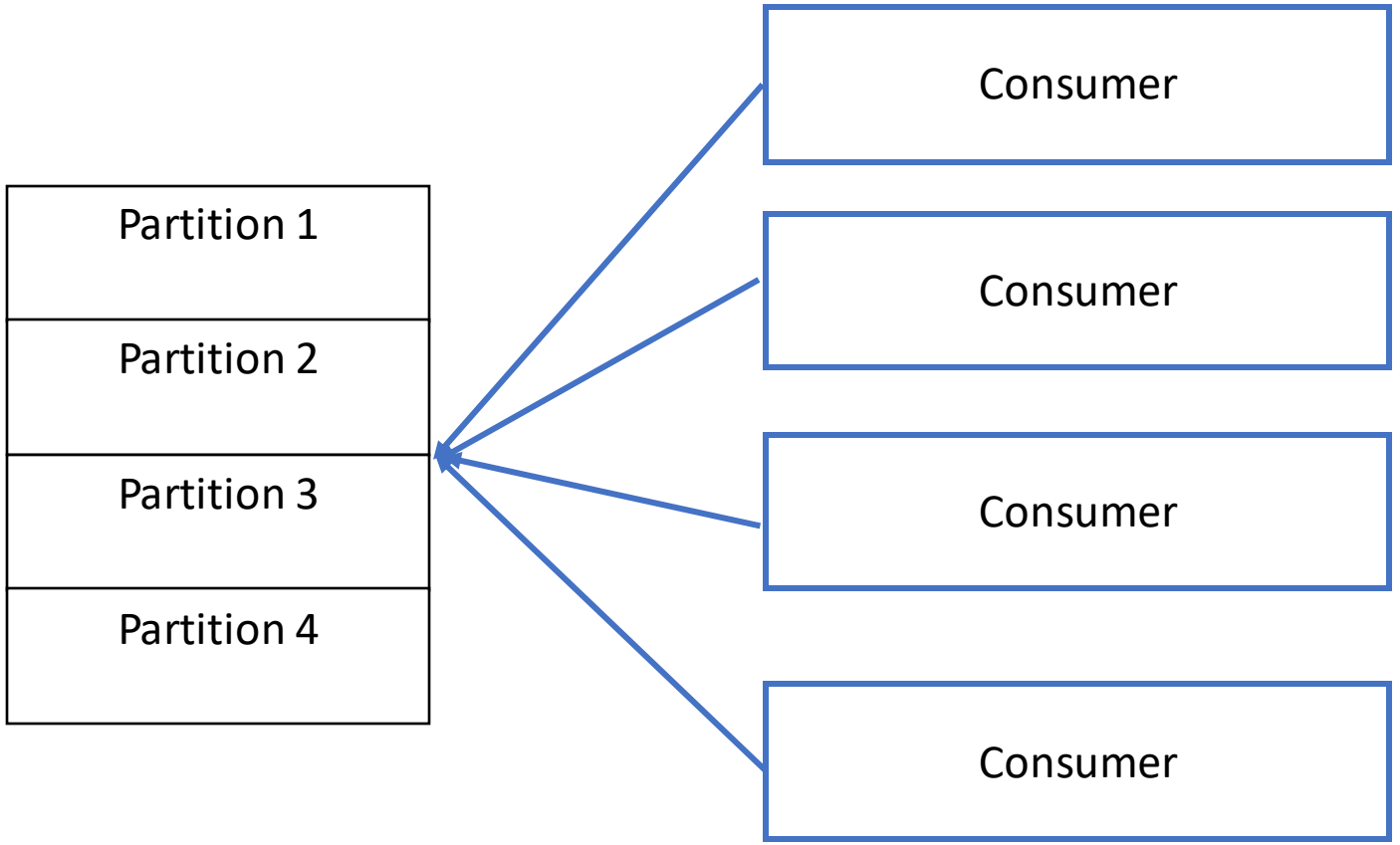
Consumers



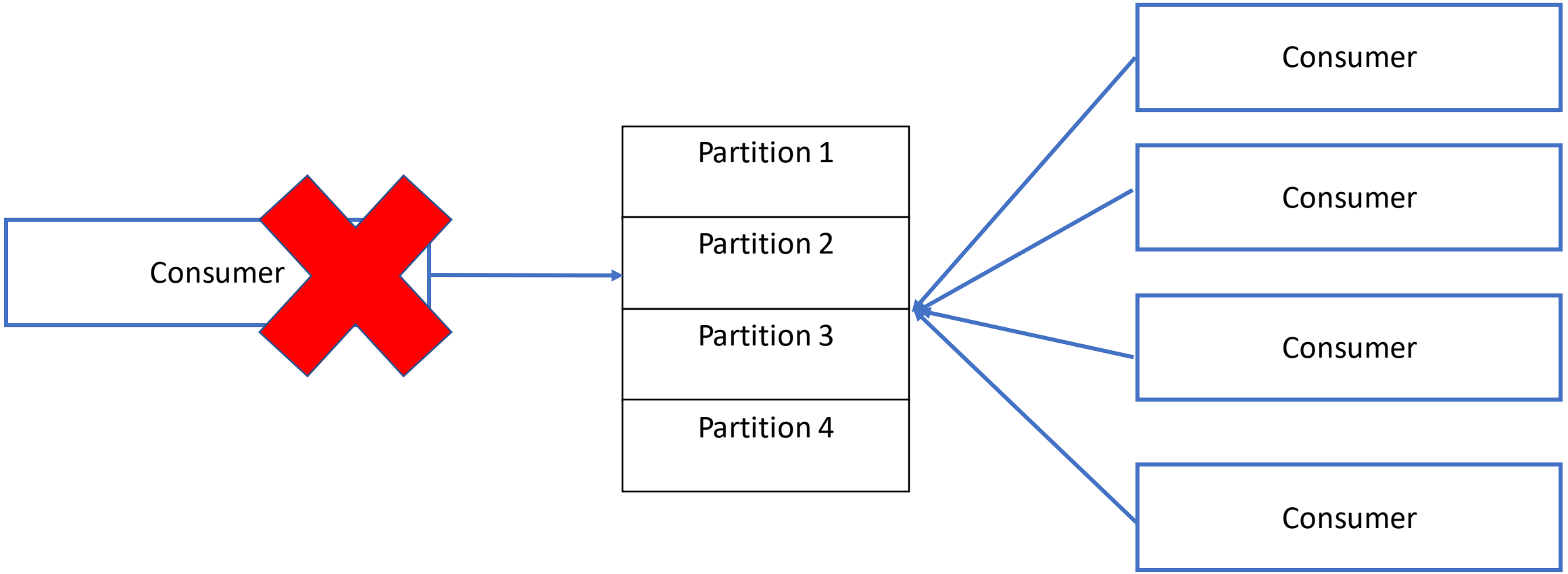
Consumers


Consumer
Partition 1
Partition 2
Partition 3
Partition 4
Consumer

Consumers


Partition 1
Partition 2
Partition 3
Partition 4
Consumer
Consumer
Consumer
Consumer

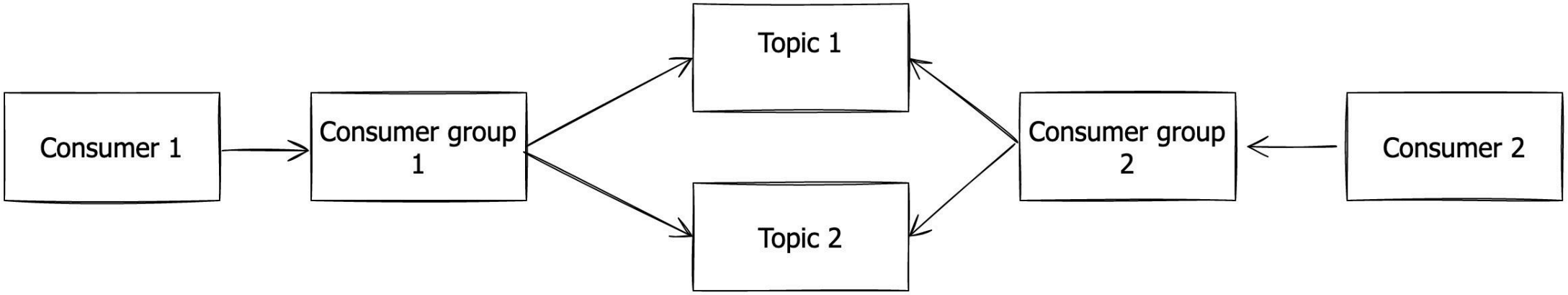
Consumers


Consumer
Partition 1
Partition 2
Partition 3
Partition 4
Consumer
Consumer
Consumer
Consumer

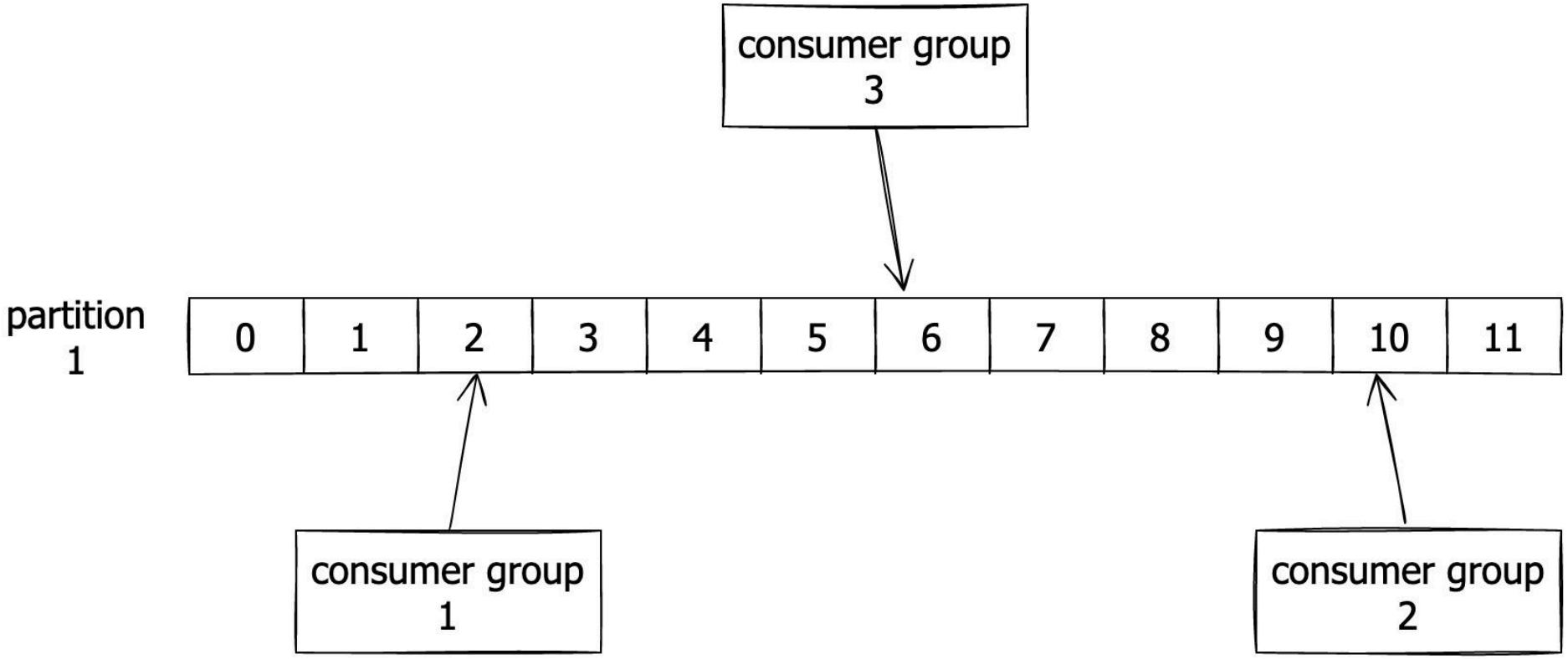
Limitation

- Within the partition the order is guaranteed
- Between the partitions the order is unpredictable

Consumer groups



Several consumers can read 1 topic independently



Low-level and high-level consumption

Low level consumption (without consumer group)

kcat -C -b localhost:9092 -t test -p 0 -o beginning

```
❯ kcat -C -b localhost:9092 -t rebalancing-demo -p 0 -o beginning -e
1
2
% Reached end of topic rebalancing-demo [0] at offset 6: exiting

~
❯ kcat -C -b localhost:9092 -t rebalancing-demo -p 0 -o beginning -e
1
2
% Reached end of topic rebalancing-demo [0] at offset 6: exiting
```

Low-level and high-level consumption

High-level consumption (with consumer group)

kcat -G test-group -b localhost:9092 test

```
❯ kcat -G rebalancing -b localhost:9092 -e rebalancing-demo
% Waiting for group rebalance
% Group rebalancing rebalanced (memberid rdkafka-b32f75c2-c700-4
1
2
% Reached end of topic rebalancing-demo [0] at offset 8
% Reached end of topic rebalancing-demo [2] at offset 0
% Reached end of topic rebalancing-demo [1] at offset 0: exiting
% Group rebalancing rebalanced (memberid rdkafka-b32f75c2-c700-4
```

Last offset always stored for the combination of:

- Topic
- Consumer group
- Partition

Rebalancing

Example of partition assigning (1 consumer)

```
GROUP          TOPIC              PARTITION  CURRENT-OFFSET  CONSUMER-ID
rebalancing    rebalancing-demo 0            8               rdkafka-dee7f3
rebalancing    rebalancing-demo 1            -               rdkafka-dee7f3
rebalancing    rebalancing-demo 2            -               rdkafka-dee7f3
```

Example of partition assigning (2 consumers)

```
GROUP          TOPIC              PARTITION  CURRENT-OFFSET  CONSUMER-ID
rebalancing    rebalancing-demo 0            8               rdkafka-dee7f3
rebalancing    rebalancing-demo 1            -               rdkafka-dee7f3
rebalancing    rebalancing-demo 2            -               sarama-a0104d2
```

Rebalancing

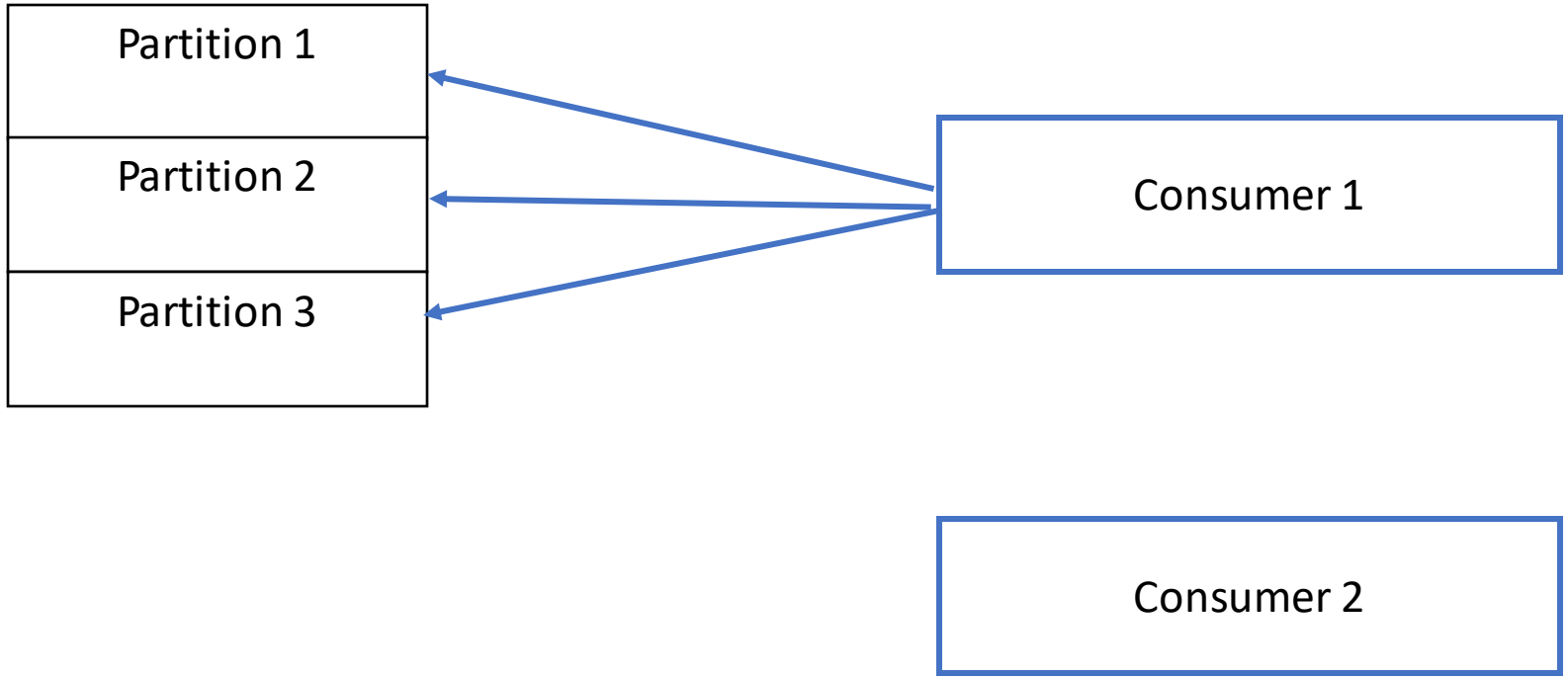
WHEN:

- New consumer is added
- Old consumer is gone

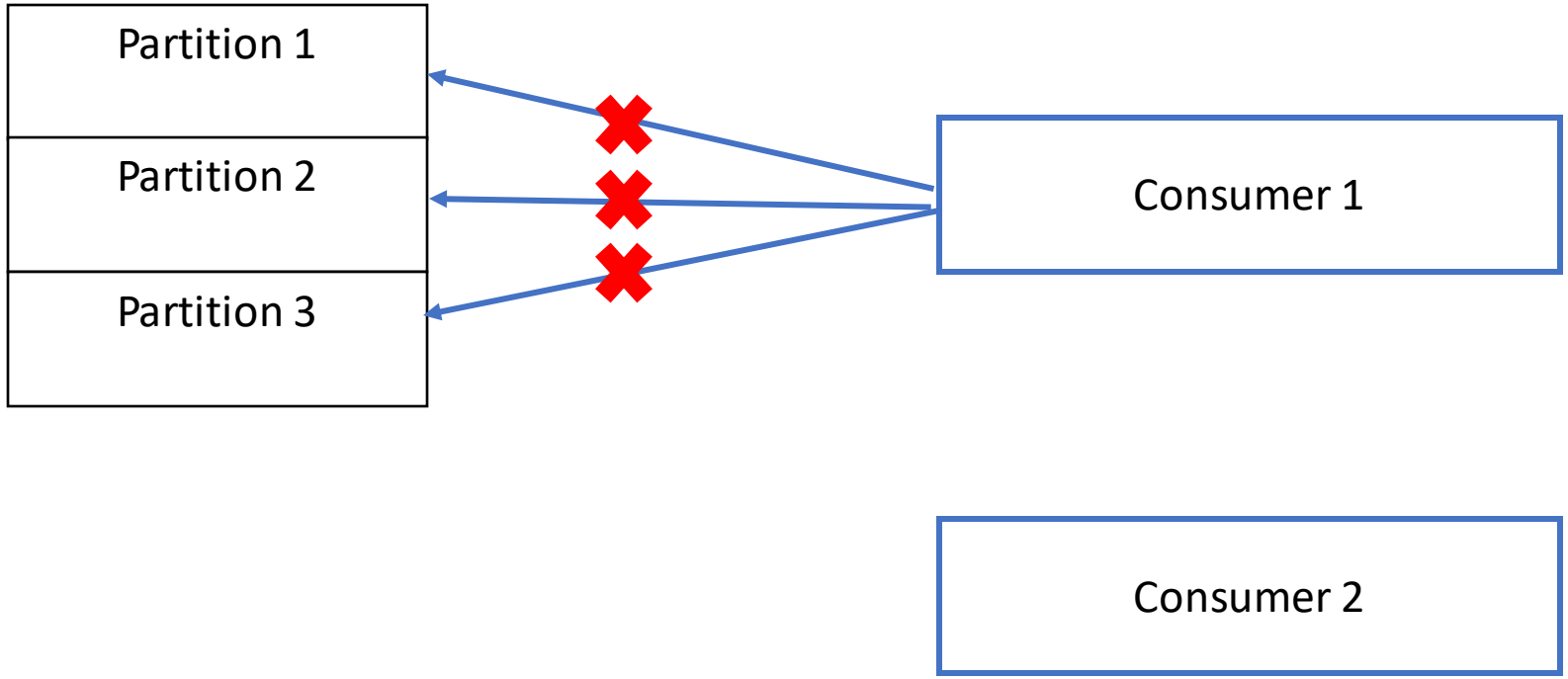
EVENTS:

- Redeployment
- Scaling up(down)
- App got stuck

Rebalancing

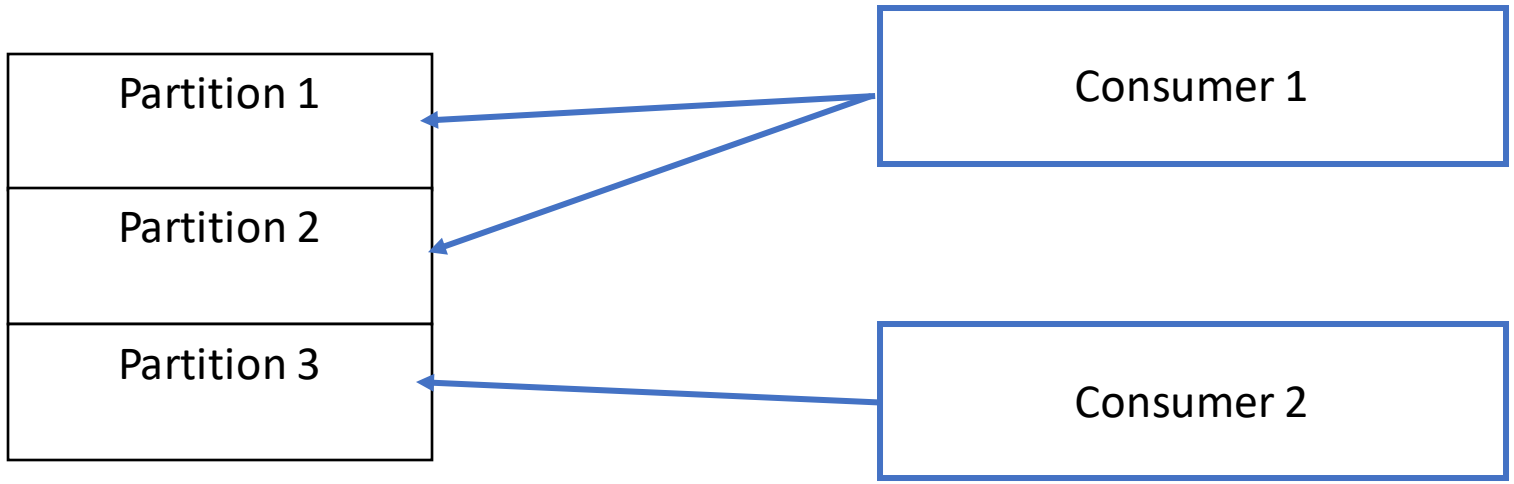


Rebalancing



Rebalancing

Rebalancing



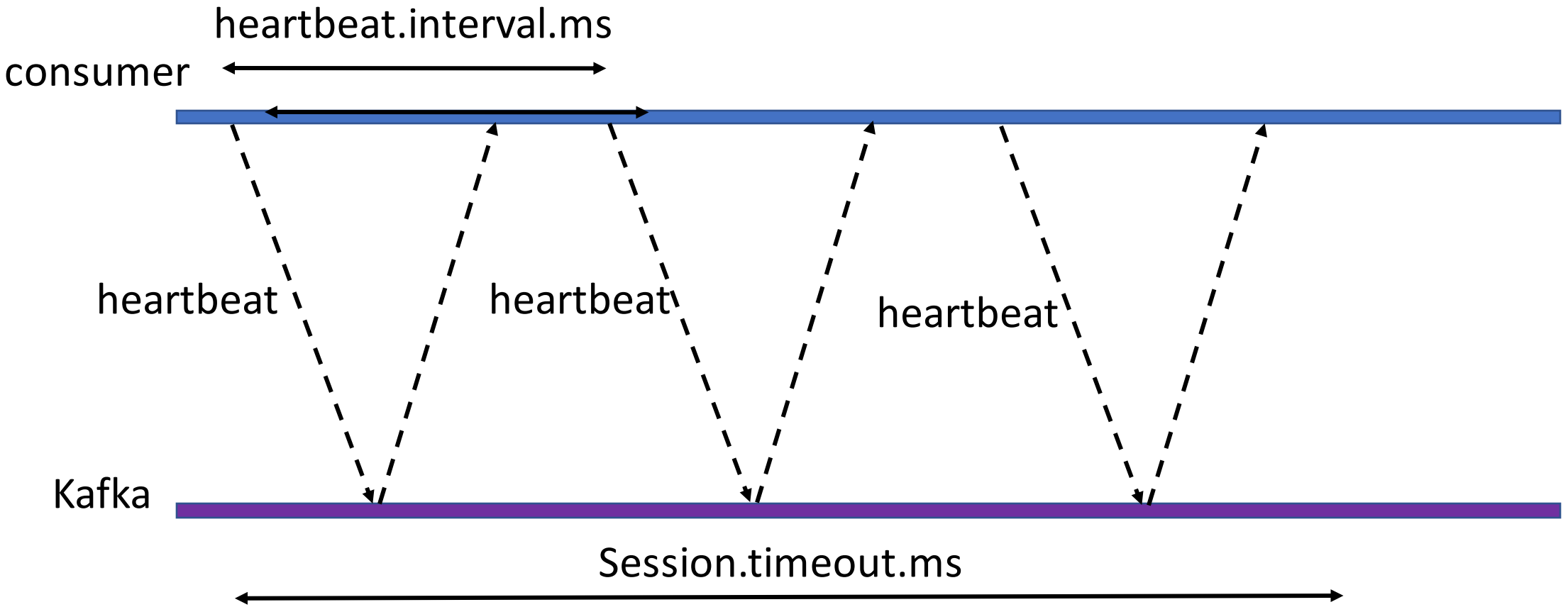
Rebalancing

Parameters: how Kafka can detect that you app's gone

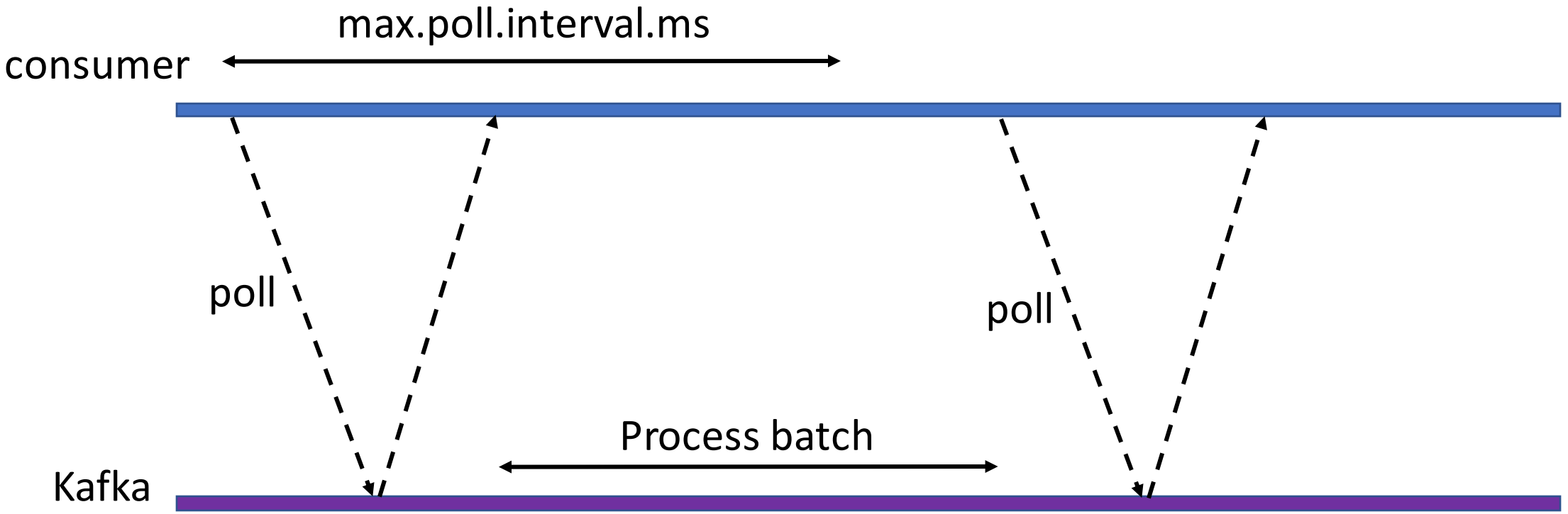
- session.timeout.ms
- kafka heartbeat.interval
- max.poll.interval.ms

- group.initial.rebalance.delay.ms

Rebalancing



Rebalancing



Consumer group status

As usual

```
GROUP          TOPIC              PARTITION  CURRENT-OFFSET  CONSUMER-ID
rebalancing    rebalancing-demo 0          8               rdkafka-dee7f3
rebalancing    rebalancing-demo 1          -               rdkafka-dee7f3
rebalancing    rebalancing-demo 2          -               rdkafka-dee7f3
```

When rebalancing

```
Warning: Consumer group 'rebalancing' is rebalancing.
```

Rebalancing

Consume actions (Shopify/sarama)

```
func (m Message) ConsumeClaim(session sarama.ConsumerGroup
    for {
        select {
        // 1. Read
        case message, ok := <-claim.Messages():
            if !ok : nil ↗
            // 2. Handle
            fmt.Printf( format: "topic: %q", message.Topic)
            // 3. Commit
            session.MarkMessage(message,  metadata: "")
            session.Commit()
        }
    }
```

Rebalancing

What should you do after rebalancing event is fired?

1. Stop reading messages
2. Consider stop handling
3. Commit messages (that are already handled)

Where?

ConsumeGroupHandler.Cleanup (Shopify/sarama)

Manual Commit

Auto-commit

Shopify/sarama

```
config.Consumer.Offsets.AutoCommit.Enable = true
config.Consumer.Offsets.AutoCommit.Interval = 5 * time.Second
```

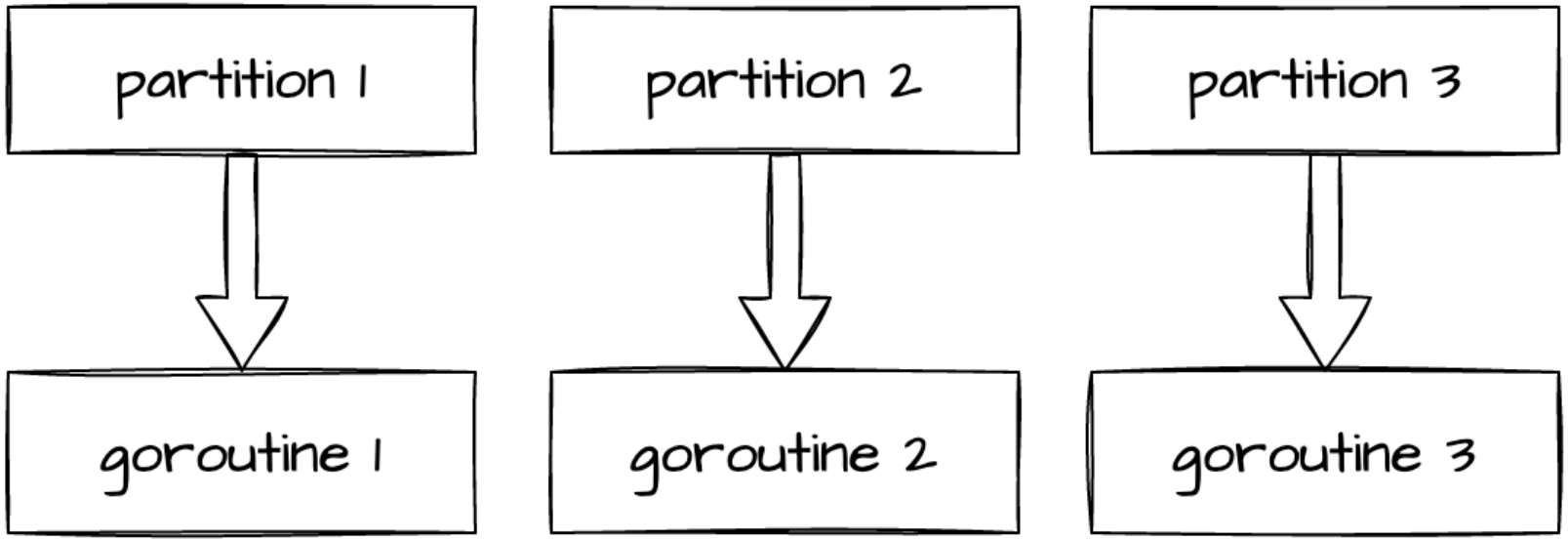
Why you don't like it?

- Commit regardless of handling status

Manual Commit

Concurrent handling

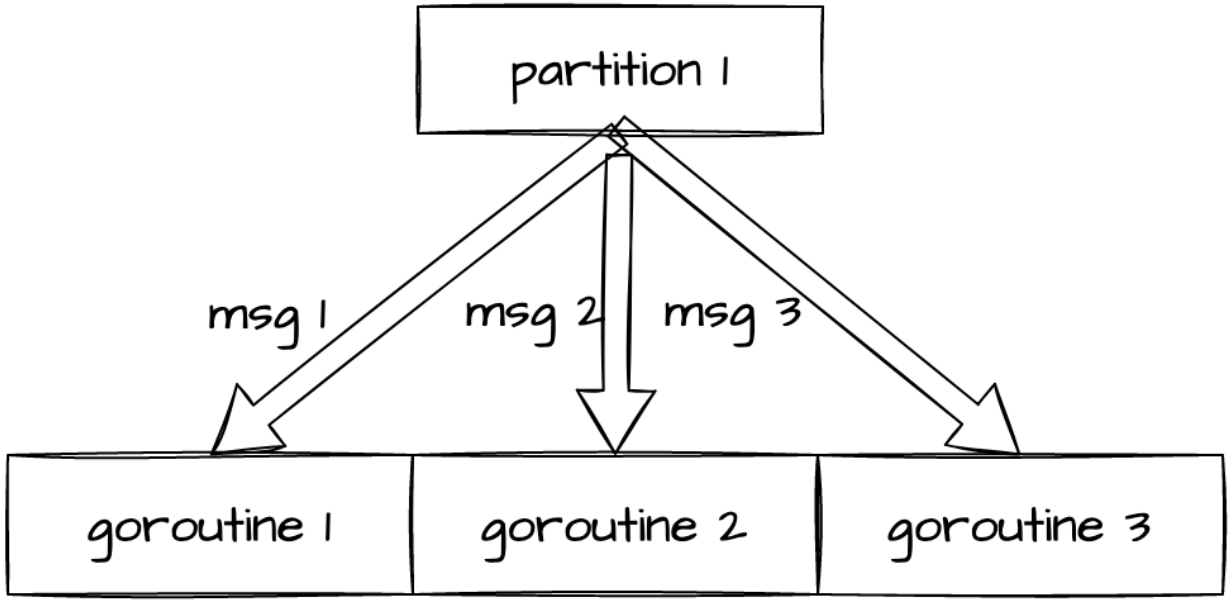
- 1 goroutine per 1 partition



Manual Commit

Can we increase the degree of parallelism?

- channelBufferSize

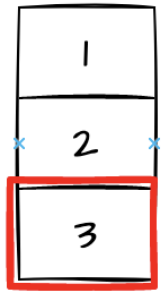


Manual commit

What is the correct order of handling?

- Unsorted
- No way to commit properly

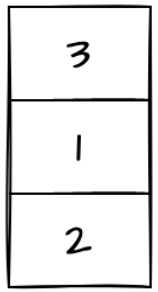
- Sorted
- Commit every n-th message



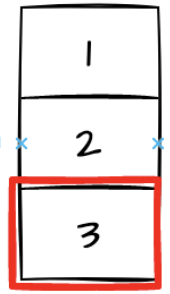
Manual commit

What is the correct order of handling?

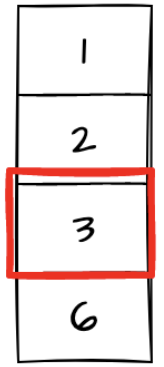
- Unsorted
- No way to commit properly



- Sorted
- Commit every n-th

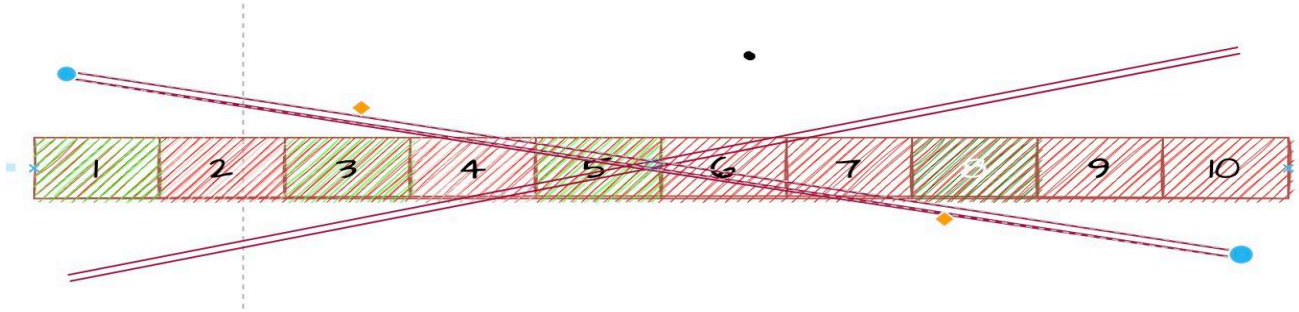


- Sorted w/o gaps

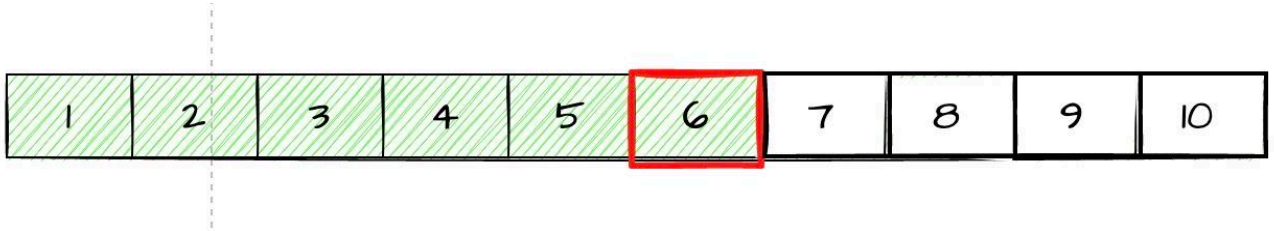


Committed messages

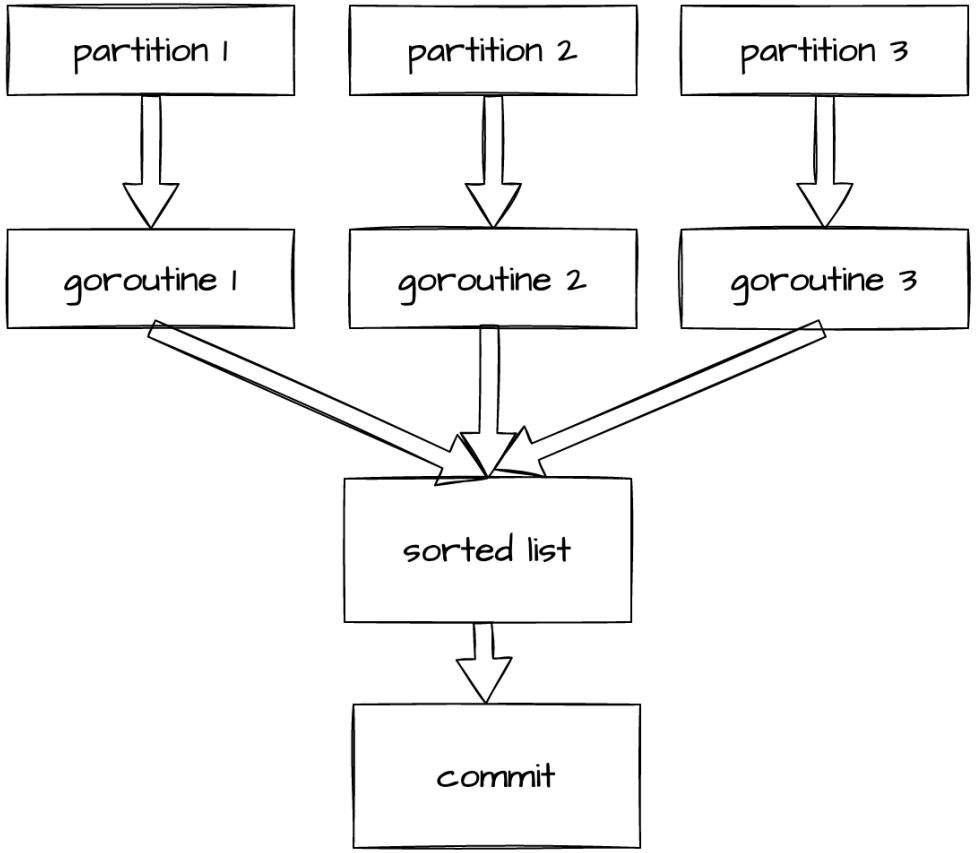
- You cannot mark only few messages as committed

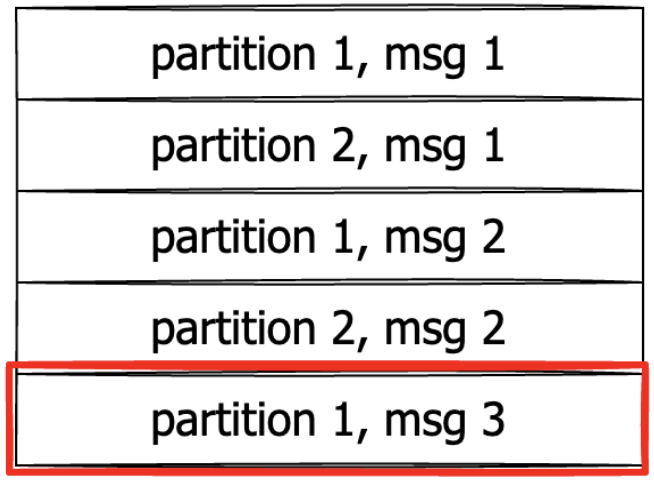


- You can mark only one message. All the previous ones will be considered as committed



Manual commit





Producing in batches

segmentio/**kafka-go**

```
writer := &kafka.Writer{
    Async:        false,
    Addr:         kafka.TCP( address...: "localhost:9092"),
    Topic:        "test",
    Balancer:     &kafka.LeastBytes{},
    WriteTimeout: writeTimeout,
    BatchSize:    200,
    BatchTimeout: 1 * time.Second,
}
```

Producing in batches

segmentio/**kafka-go**

```
var messages []kafkago.Message

writer.WriteMessages(ctx, messages...)
```

Producing in batches

```
w := &kafka.Writer{
        Addr:     kafka.TCP("localhost:9092", "localhost:9093", "localhost:9094"),
        Topic:   "topic-A",
        Balancer: &kafka.LeastBytes{},
}

err := w.WriteMessages(context.Background(),
        kafka.Message{
                Key:   []byte("Key-A"),
                Value: []byte("Hello World!"),
        },
```

Wrapping up...

- Rebalancing
  - Consider rebalancing as part of your consumption process
  - Configure session.timeout wisely
  - Commit all processed messages during the rebalancing event

- Manual commit
  - Use manual commit rather than the automatic one
  - Use only the partitions for paralellism

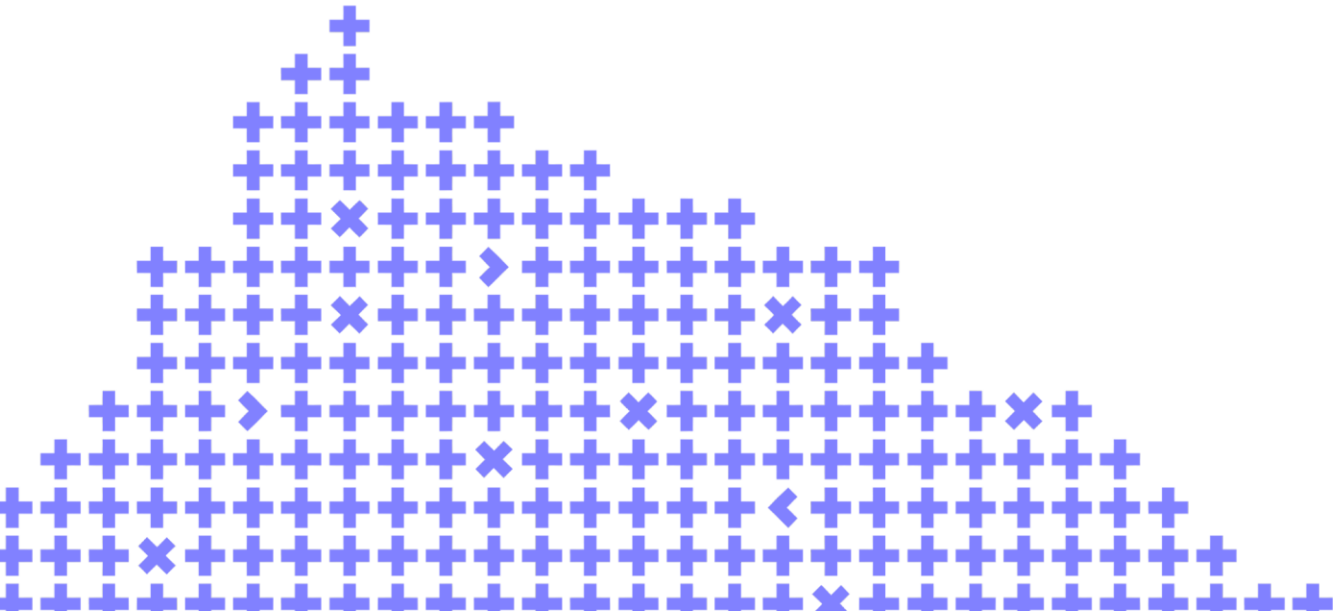
- Producing in batches
  - Carefully use the defaults

About the author

- Denis Filippov
- https://t.me/filippov_sequel
- These slides: http://bit.do/kafka_highload_2022

Leave your feedback!

You can rate the talk and give a feedback on what you've liked or what could be improved



High Load Armenia

Co-organizer

Yandex